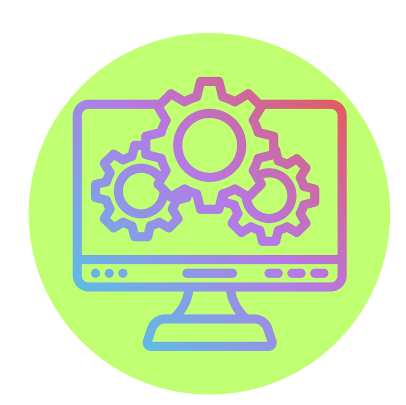
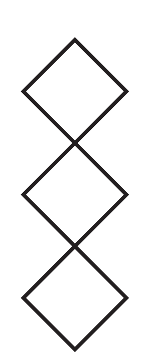
2024



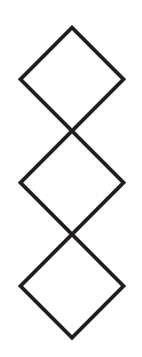






NAJIB EL KHADIR
INGENIEUR LOGICIEL

PORTFOLIO



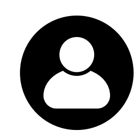
najibxy.github.io/

TABLE DES MATIERES





Qui suis-je ?



github.com/NajibXY
najibxy.github.io
linkedin.com/in/najib-el-khadir-403b49b8/

Je m'appelle Najib, je suis un développeur logiciel et un passionné de sciences et de jeux vidéo. Je travaille depuis plus de deux ans en tant qu'Ingénieur Logiciel dans le domaine de l'immobilier social. Également, je réalise plusieurs projets sur mon temps libre afin d'orienter ma carrière vers ce qui est ma vocation ; et ce qui sera mon prochain challenge : le développement de jeux vidéo.

Après avoir obtenu mon BAC en Sciences Expérimentales mention Très Bien au Maroc, j'ai poursuivi mon cursus en obtenant un DUT Informatique suivi d'un parcours à l'Université Claude Bernard Lyon 1, se soldant par l'obtention d'un Master Informatique Spécialisé en Intelligence Artificielle.

Fort de mes nombreuses expériences professionnelles, d'abord en tant que développeur d'un Intranet lors de mon stage de DUT, ensuite en tant que stagiaire chercheur au LIRIS sur le sujet de l'apprentissage profond par renforcement et enfin en tant qu'ingénieur R&D sur un logiciel à échelle industrielle leader chez les bailleurs sociaux ; et travaillant sur plusieurs projets personnels autour du domaine du jeu vidéo, de la visualisation et de l'intelligence artificielle, je souhaite m'inscrire au Master 2 Ingénierie du Jeu vidéo à Gamagora en reprise d'études afin de solidifier mes connaissances, d'obtenir des compétences primordiales et de développer mon réseau. Et ce, dans le but de concrétiser mon projet professionnel s'articulant autour des applications de l'intelligence artificielle au domaine du jeu vidéo, ainsi que de donner vie à mes nombreux projets vidéoludiques.

Tous les projets cités dans ce portfolio sont consultables sur mon GitHub.




AUTOUR DU JEU VIDEO ET DE L'IA

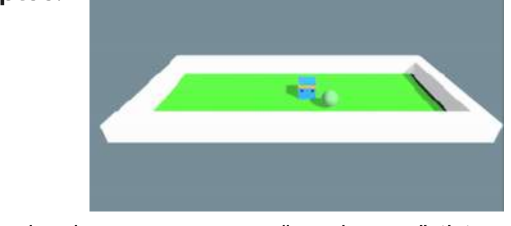



CarController : Apprentissage par renforcement sur Unity avec ML-Agents

Afin de monter en compétences sur **Unity** et en **Machine Learning**, j'ai décidé de monter un projet de diverses expérimentations dans des environnements d'**apprentissage par renforcement** que je conçois en parallèle dans le moteur Unity.
Cela implique plusieurs axes de développement :

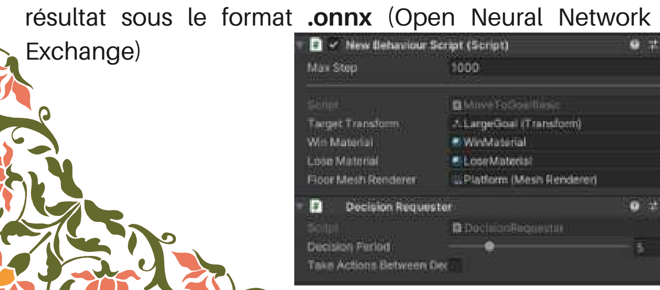
- Concevoir des Scènes sur Unity.
- Concevoir des Controllers et des Agents apprenants sur Unity.
- Concevoir et dérouler des scénarios d'apprentissage automatique grâce à la bibliothèque ML-Agents.

Ainsi qu'un objectif à court terme : **développer un agent capable de contrôler une voiture et conduire sur n'importe quelle scène Unity adaptée**.



Afin de développer mon agent "conducteur", j'ai commencé par entrainer un agent sur un **environnement "Reach the Goal"**. Cela a impliqué :

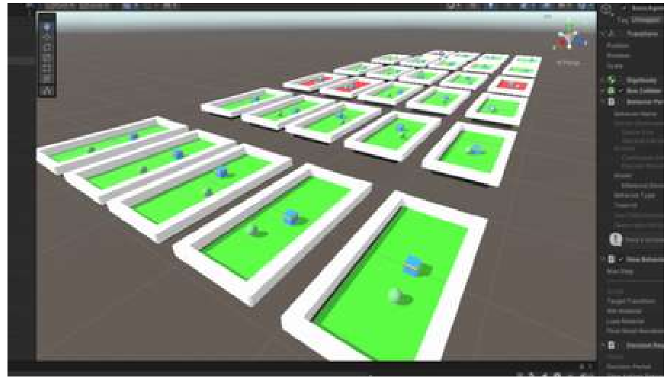
- La **préparation de la scène** :
  - Conception de l'environnement dans Unity : les différents prefab, les éléments **interactifs**
  - Conception la **logique de récompense** et d'apprentissage et son implémentation dans le script de l'agent grâce à l'interface de la bibliothèque ML-Agents **Agent**.
  - Documentation à propos des différents **composants relatifs à l'apprentissage** et implémentation de ceux-ci
- La **préparation de l'apprentissage** : en clonant l'environnement dans Unity et en rajoutant de l'aléatoire dans les points d'apparition, j'ai pu **entrainer plusieurs agents sur le même objectif et accélérer** la convergence vers une stratégie gagnante.
- Lancement de l'apprentissage. Observations. Exports du résultat sous le format **.onnx** (Open Neural Network Exchange)

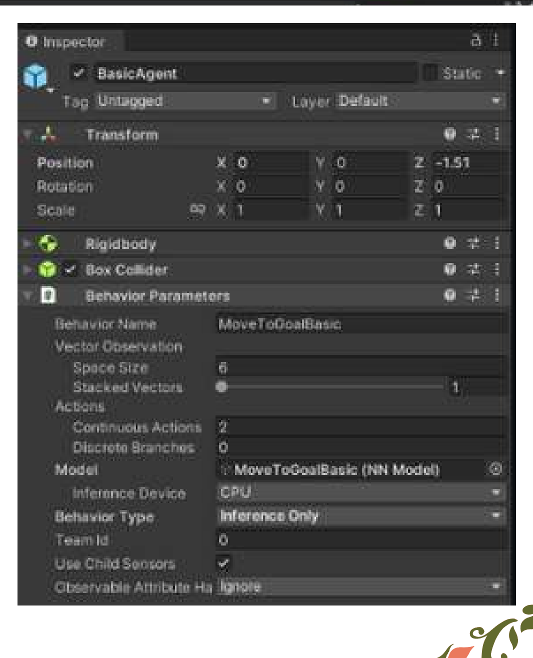




Technologies utilisées :

- **C#, Unity 2022.3.14.f1, package ML-Agents** : pour la conception et le développement des agents et des environnements d'entrainement
- **Python 3.9, PyTorch**, Conda avec la bibliothèque Python **ML-Agents version 0.30.0** pour l'entraînement des agents

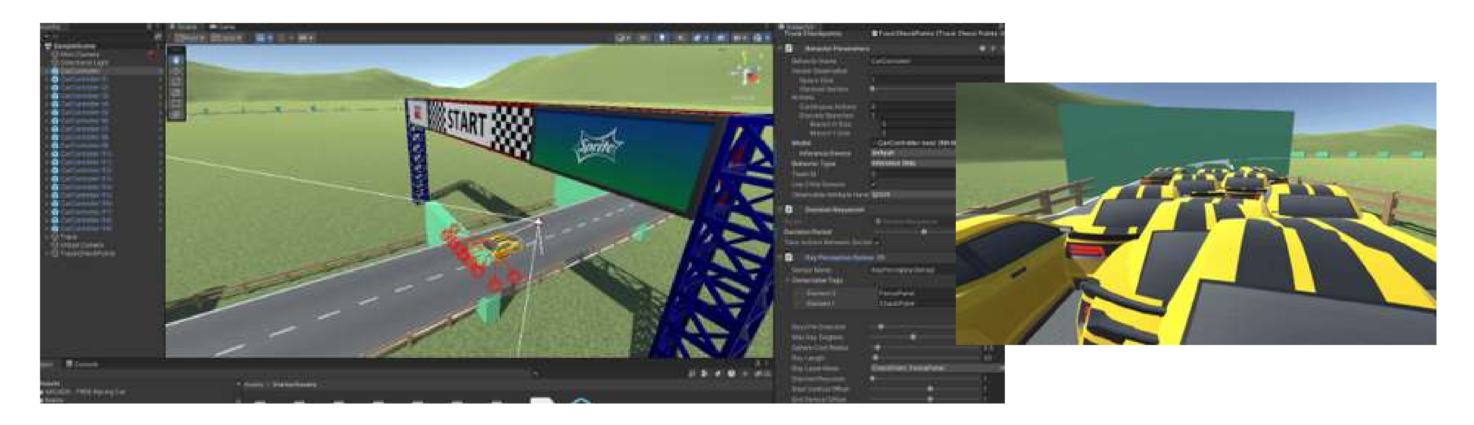




CarController : Apprentissage par renforcement sur Unity avec ML-Agents (2)

Ensuite, j'ai appliqué la même logique de conception en construisant un **environnement adapté pour l'apprentissage de la conduite** :

- Conception et réalisation de la **voiture d'entrainement**, de **ses différents meshs et composants**
- Conception et implémentation de l'**agent contrôleur**, capable de **freiner, accélérer, tourner à gauche et tourner à droite.**
- Conception et réalisation d'un **circuit** et d'un **système de barrières et de checkpoints** autour duquel s'articule le **mécanisme d'apprentissage**





Débuts de l'appentissage



Apprentissage avancé

Après plus d'un million d'étapes d'apprentissage, les agents **se déplacent de manière fluide dans le circuit** et ont **appris à faire des tours**. À ce stade, nous pouvons **exporter un modèle** entrainé à plusieurs stades d'apprentissages (pouvant par exemple correspondre au niveau de difficulté de l'agent apprenant)

Retour sur l'objectif initial

- L'objectif actuel étant de développer un agent capable de conduire sur n'importe quel circuit sur Unity (tant que le script de l'agent expose certaines fonctions nécessaires du package ML-Agent), il faut que je réalise l'apprentissage sur plusieurs types de circuits en même temps afin d'avoir un agent résultant adaptatif.
- Affaire à suivre !

En savoir plus ou contribuer au projet :
https://github.com/NajibXY/Unity-ML-Agents/

RayOfLife : Simulation interactive du jeu de la vie en C++

Le jeu de la vie est un des exemples les plus marquants en matière d'**IA développementale** par le contraste entre la **simplicité des règles initiales imposées aux agents et la complexité des comportements émergents.** Cela reste un cas d'école pour tout ce qui touche aux comportements émergents et aux **systèmes intelligents bio-inspirés** ; ainsi qu'une introduction visuelle fascinante vers les courants de l'IA qui sont un peu moins à la mode de nos jours.
De plus, ce projet était l'occasion parfaite pour en apprendre plus sur l'utilisation d'une des bibliothèques dédiées au développement de jeu-vidéo les plus en vogue du moment : **Raylib**



Technologies utilisées :

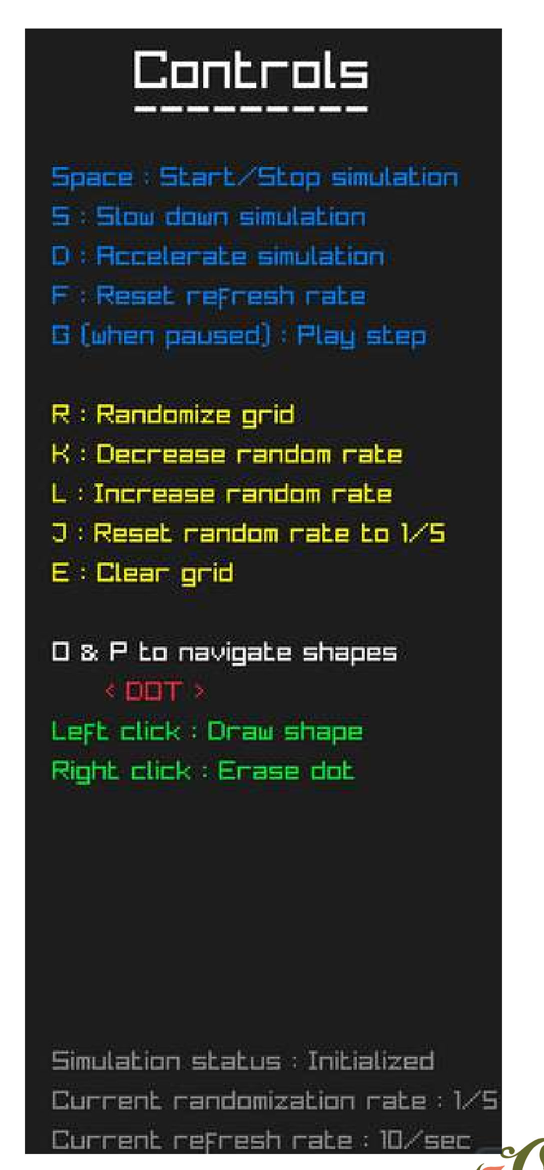
- C++14
- Librairie Raylib pour le développement de jeu vidéo

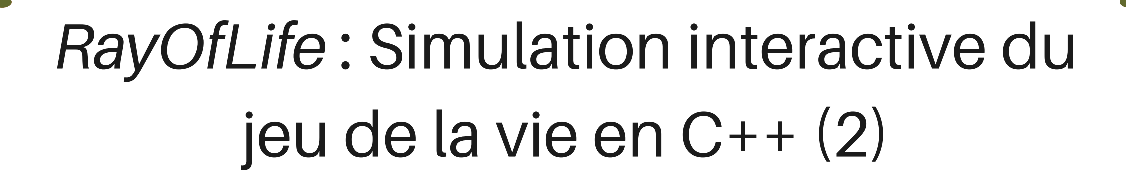
Règles de la simulation

- **Sous-population** : Si la cellule est en vie et a moins de deux voisins vivants, elle meurt à la prochaine itération
- **Stase** : Si la cellule est en vie et a un nombre de voisins vivants égal à 2 ou 3, elle reste en vie à la prochaine itération
- **Sur-population** : Si la cellule est en vie et a plus de trois voisins vivants, elle meurt à la prochaine itération
- **Reproduction** : Si la cellule n'est pas active et a un nombre de voisins vivants égal à 3, elle vit à la prochaine itération

Fonctionnalités implémentées

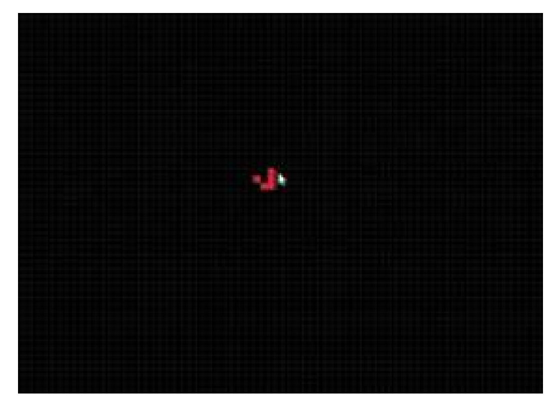
- **Randomisation** de la grille (Touche R)
- **Nettoyage** de la grille (Touche E)
- **Accélération et Ralentissement** de la simulation (Touches S D & F)
- **Mettre en pause / Reprendre la simulation** (Touche Espace)
- **Jouer une étape** (Touche G)
- Modifier le **taux de la randomisation** (Touches K L & J)
- **Ajouter manuellement des cases ou des patterns prédéfinis** (clics de la souris)
- Naviguer entre les formes à dessiner (Touches O & P)



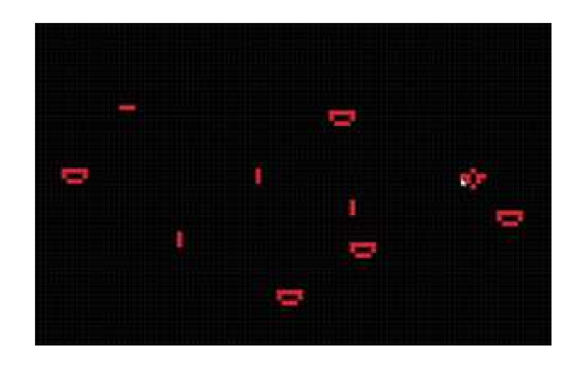


RayOfLife : Simulation interactive du jeu de la vie en C++ (2)

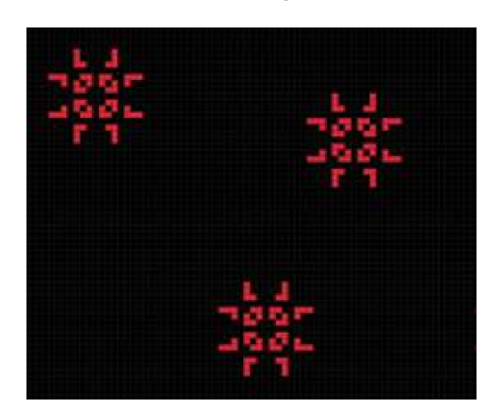
Patterns et oscillateurs implémentés



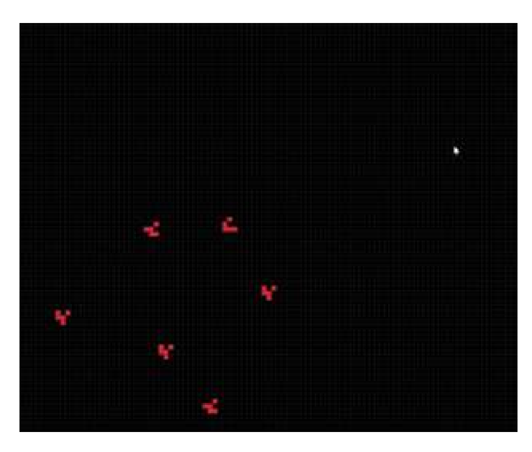
Le point (pour des formes customisés)



Le Blinker (décliné sous plusieurs formes)



Le Pulsar



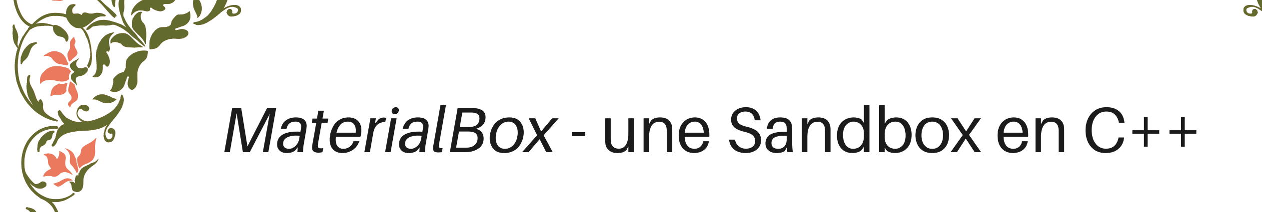
Le Glider



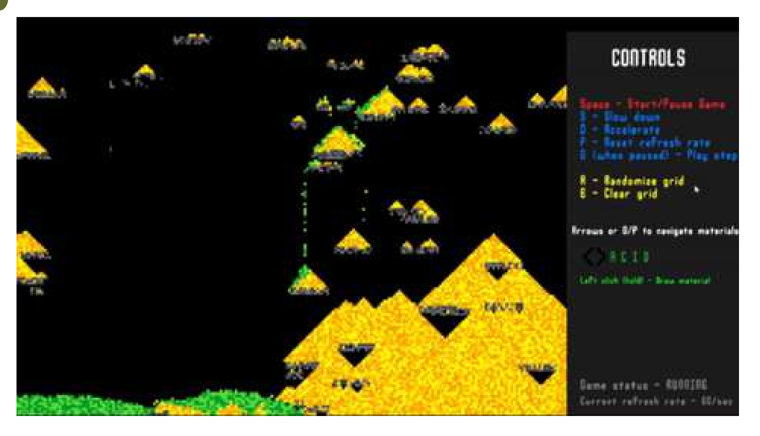
Le Gosper Glider Gun

Note :
Ce projet, par sa projection dans le temps et ses mécanismes interactifs, a pour but d'être utilisé librement comme une introduction à l'IA pour des collégiens ou autres apprenants.

En savoir plus ou contribuer au projet :
https://github.com/NajibXY/Game-Of-Life-using-CPP-and-Raylib



MaterialBox - une Sandbox en C++



Dans le cadre de mon apprentissage de l’utilisation de la bibliothèque Raylib et toujours sur le thème des automates cellulaires, j'ai repris mon projet de simulation de jeu de la vie et j'en ai développé un autre projet : celui d'une sandbox interactive où l'utilisateur peut jouer avec différents matériaux.

Technologies utilisées :

- C++14
- Librairie Raylib pour le développement de jeu vidéo

Règles du jeu

- L'utilisateur peut **dessiner** en maintenant le bouton gauche de la souris **plusieurs matériaux.**
- L'utilisateur peut **choisir le matériau** à dessiner en suivant les indications sur la fenêtre des contrôles.
- Chaque matériau à ses **règles d'activités précise**s et **interagit avec les autres matériaux**.

Contrôles implémentés

- **Randomisation de la grille** (Touche R) : la randomisation ajoute des particules de sable placées de manière aléatoire et des structures de pierre générées aléatoirement dans la fenêtre de dessin.
- **Nettoyage** de la grille (Touche E)
- **Accélération et Ralentissement** du jeu (Touches S D & F)
- **Mettre en pause / Reprendre le jeu** (Touche Espace)
- **Jouer une étape** de rafraîchissement (Touche G)
- **Modifier le matériau choisi** (Touches O/P ou Gauche/Droite ou Clics sur les boutons < et >)

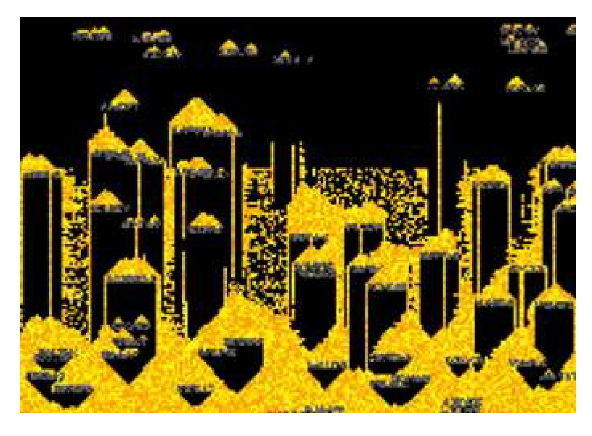






MaterialBox - une Sandbox en C++ (2)

Matériaux implémentés

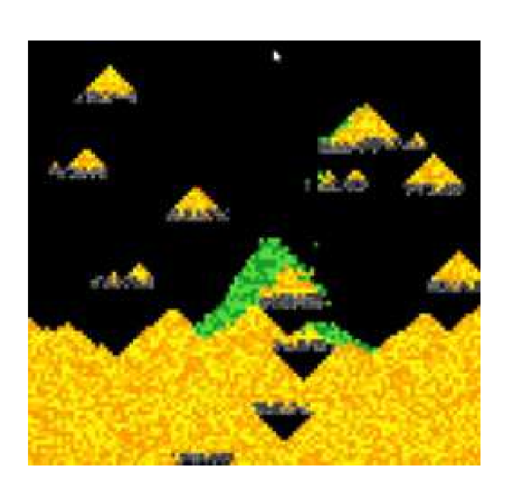


Le sable

- Matériau de base de la simulation
- Se propage comme du sable dans le monde réel
- Se fait bloquer par la pierre et se faufile partout où il peut
- Se dissout progressivement dans l'acide

La pierre

- Forme des blocs statiques sur la grille du dessin
- Bloque le passage du sable
- Lors d'une randomisation du canva, des structures de pierre générées aléatoirement sont placées
- Se dissout progressivement au contact de l'acide



L'acide

- Se propage plus facilement que le sable
- Ronge progressivement la pierre
- Dissout progressivement le sable
- Forme des flaques visuellement imprévisibles
- Des règles d'aléatoire simples rendent le comportement de ce matériau particulièrement satisfaisant à observer

Le void

- Des cases vides peuvent être remises en sélectionnant le matériau "Empty"
- Cela agit sur la simulation indépendamment de si celle-ci est en pause ou non
- Il est d'ailleurs assez fun de mettre en pause la simulation, d'ajouter du vide, puis de relancer la simulation et observer ce qui se passe.

En savoir plus ou contribuer au projet :
https://github.com/NajibXY/MaterialBox---A-Sandbox-using-CPP-and-Raylib

Implémentation d'un algorithme d'apprentissage par renforcement profond dans l'environnement VizDoom



Durant ma deuxième année en master informatique spécialisé en Intelligence Artificielle, j'ai eu l'occasion d'entendre parler d'une bibliothèque Python fournissant des environnements d'entraînement pour agents apprenants : VizDoom. Dans le cadre donc de mes projets personnels, j'ai entrepris d'implémenter un **algorithme d'apprentissage par renforcement profond et de le tester sur des environnements de VizDoom**.

Technologies utilisées :

- Python 3.12, PyTorch, VizDoom, Conda

Résumé de l'implémentation

(disponible sur le repo du projet)

Paramètres de l'environnement

- Définition des configurations VizDoom à charger.
- L'environnement étant **perçu** par l'agent comme un **ensemble de pixel**, il faut définir une résolution qui sera utilisée dans VizDoom et une fenêtre de répétition.

Classe QNN

- Cette classe est le cœur de l'**algorithme d'apprentissage profond**.
- Elle permet de créer un **réseau de neurones convolutif avec 1 couche d'entrée, 1 couche de sortie et 2 couches principales**.
- Elle permet à l'agent de **sélectionner la meilleure action à une étape donnée**, de **jouer une étape** et d'**adapter son comportement** en fonction de la Replay Memory.

Replay Memory

- Une classe Replay Memory est implémentée.
- Cette classe est utilisée pour **stocker les transitions récentes** (état source, action, état résultat, récompense).
- Cette classe permet aussi de **tracer les actions terminales** (mettant fin à l'épisode courant).

Fonctions utiles

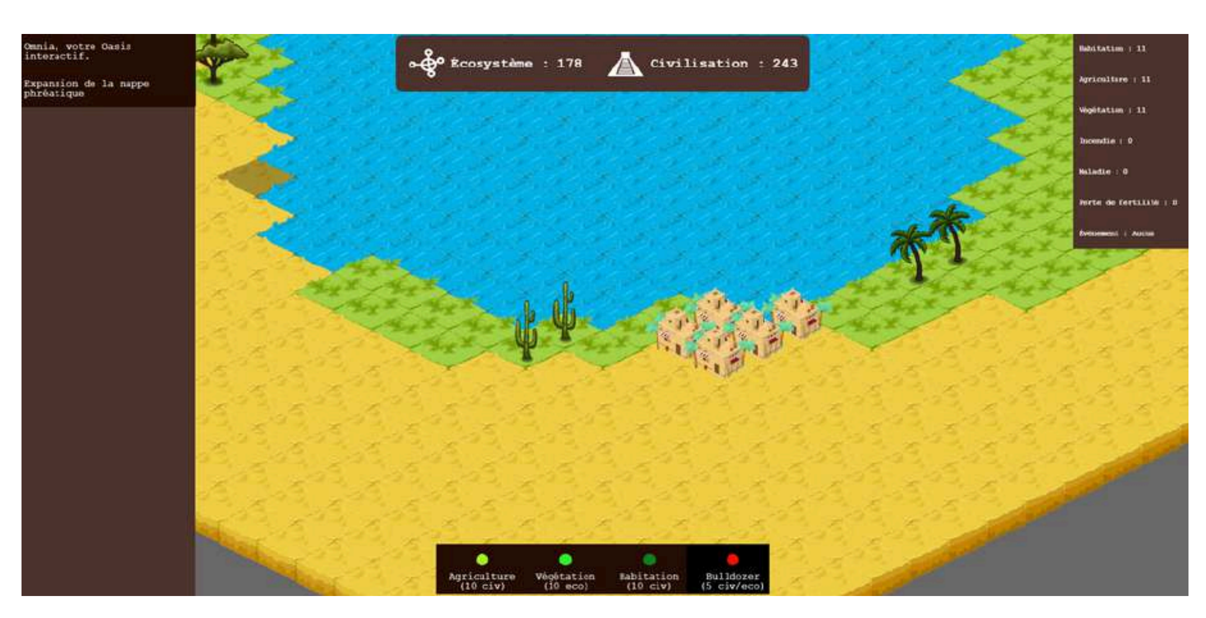
- Il y a également des fonctions permettant le **traitement de l'image courante** de la fenêtre VizDoom, d'**initialiser la simulation** et de **connaitre son état**.
- Une fonction est aussi dédiée au déroulement d'**épisodes de démo** une fois l'apprentissage terminé. Cela nécessite de changer les paramètres adéquats dans la boucle principale du script.
- La partie FLAGS de la fonction principale permet aussi de **tuner les paramètres d'entraînement** (nombre d'épisodes, nombre d'itérations par épisode, taille du batch, taux d'apprentissage, etc.). N'hésitez pas à **expérimenter avec cette partie.**

Résultats

- Sur un **environnement basique** où l'agent doit se déplacer de gauche à droite et tirer sur la cible, les résultats se sont **avérés satisfaisants** au bout de 40 minutes d'entrainement.
- Sur un **environnement plus complexe** où l'agent doit défendre le milieu, la stratégie adoptée par l'agent est **cohérente, mais non effective** (à observer en exemple dans le repo)

En savoir plus ou contribuer au projet :
https://github.com/NajibXY/ViZDoom-Deep-Q-Learning-implementation

OMNIA : un serious game sur la désertification



Durant ma première année de Master, Thomas Boffy, William Chazot ainsi que moi-même Najib El khadir (équipe ELCHABO) avons développé un **prototype de jeu-vidéo de catégorie serious game informatif sur le fonctionnement d'une oasis et alertant sur les problèmes de désertification en général.**

Technologies utilisées :

- JavaScript
- Bibliothèque pour le développement de jeux vidéo Phaser 3.20.1

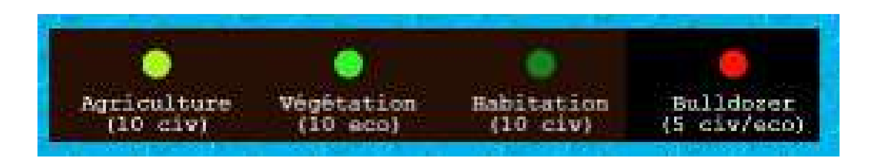


- Le jeu se présente comme un **jeu de gestion d'oasis**, où le joueur doit **préserver ses habitations et sa zone fertile du fléau de la désertification.**
- Au **centre** de l'écran, le joueur trouve **son oasis**, les cases bleues représentent l'**eau**, les vertes **la terre fertile** sur laquelle l'utilisateur peut **planter des arbres ou une unité de production agricole.** Les maisons représentent **les habitations.**

- En haut se trouve **les points d'écosystème** (la capacité à planter de la végétation) et **les points de civilisation** (la capacité à pratiquer l'agriculture et augmenter le nombre d'habitations).
- C'est l'augmentation de ses ressources qui permet d'**avancer dans le jeu.**



OMNIA : un serious game sur la désertification (2)

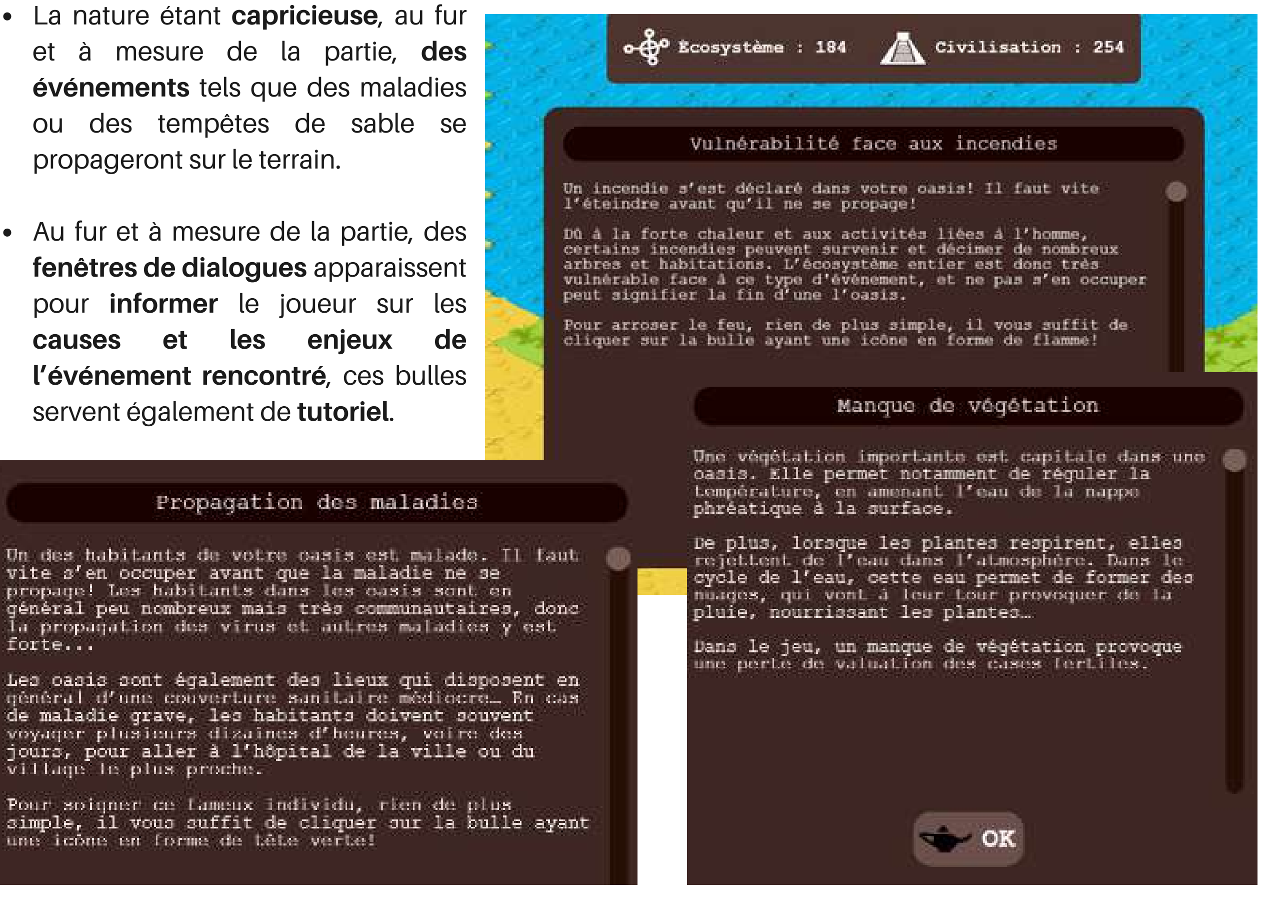


- En bas de la fenêtre de jeu se trouve la sélection des **cases à poser, ainsi que leur coût**, le bulldozer permet évidemment de détruire l'entité sur une case.

- Le joueur doit trouver **l'équilibre sur la consommation et la génération des ressources** au fur et à mesure de la partie et **des modifications que le joueur et le système auront imposées sur le terrain**.

- La nature étant **capricieuse**, au fur et à mesure de la partie, **des événements** tels que des maladies ou des tempêtes de sable se propageront sur le terrain.

- Au fur et à mesure de la partie, des **fenêtres de dialogues** apparaissent pour **informer** le joueur sur les **causes et les enjeux de l'événement rencontré**, ces bulles servent également de **tutoriel**.



- Mon rôle sur ce projet consistait en la **conception et la réalisation de la carte** de jeu (récupération des différents assets et génération de la map), ainsi que la conception et le développement de la **logique de déroulement du jeu** (état, événements aléatoires, recalcul des valeurs du jeu et du statut de la partie).

En savoir plus ou contribuer au projet :
https://github.com/NajibXY/Omnia

Checkshake (v0.4.2): Modélisation distribuée à travers un jeu stratégique

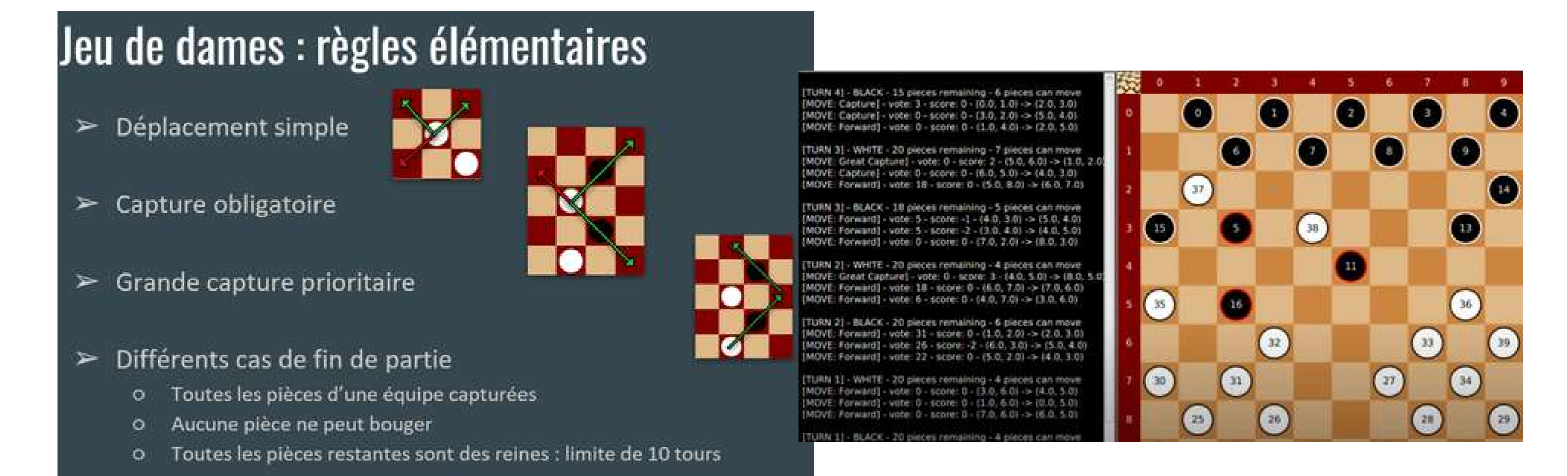
- Durant mon année de licence informatique, j'ai travaillé avec l'équipe précitée ELCHABO sur un **système multi-agents appliqué à une version modifiée du jeu de dame**s

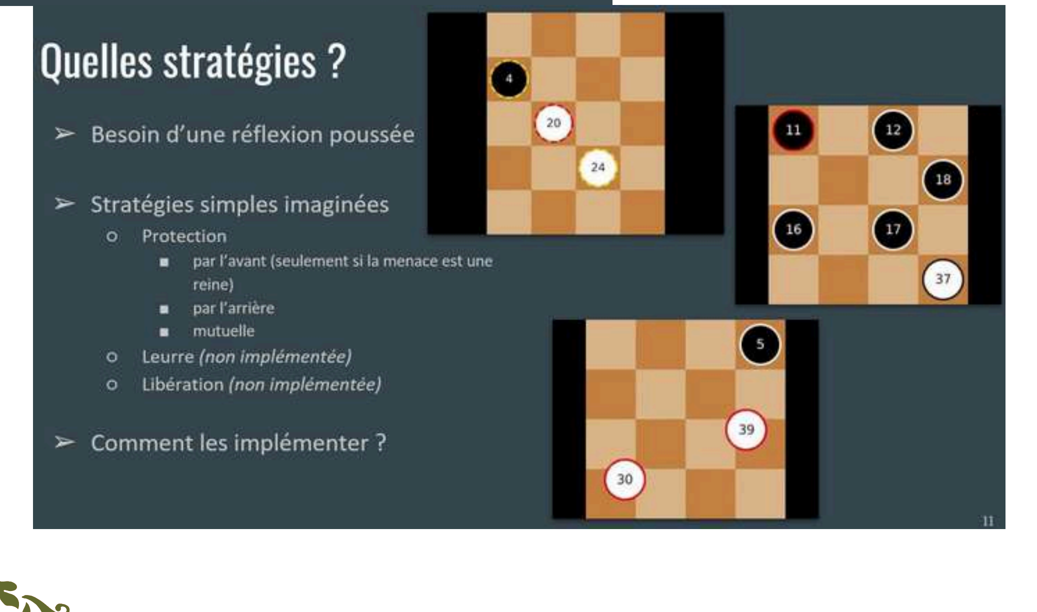
- L'objectif de ce projet était de proposer une **modélisation distribuée d'un jeu stratégique**, ici un jeu de dames, dans lequel chaque pièce devait **réfléchir individuellement puis collectivement** afin d'établir des **stratégies avec son équipe**, dans le but de **vaincre l'équipe adverse**.

- Le projet se découpait ainsi en deux sous-objectifs : le premier impliquant **la mise en place d'un jeu de dames**, ses concepts et ses règles ; et le second impliquant **la création d'un système gérant différentes stratégies, individuelles et collectives**, d'où l'importance de **la communication entre les entités du jeu.**

Technologies utilisées :

- Java 8
- Java FX



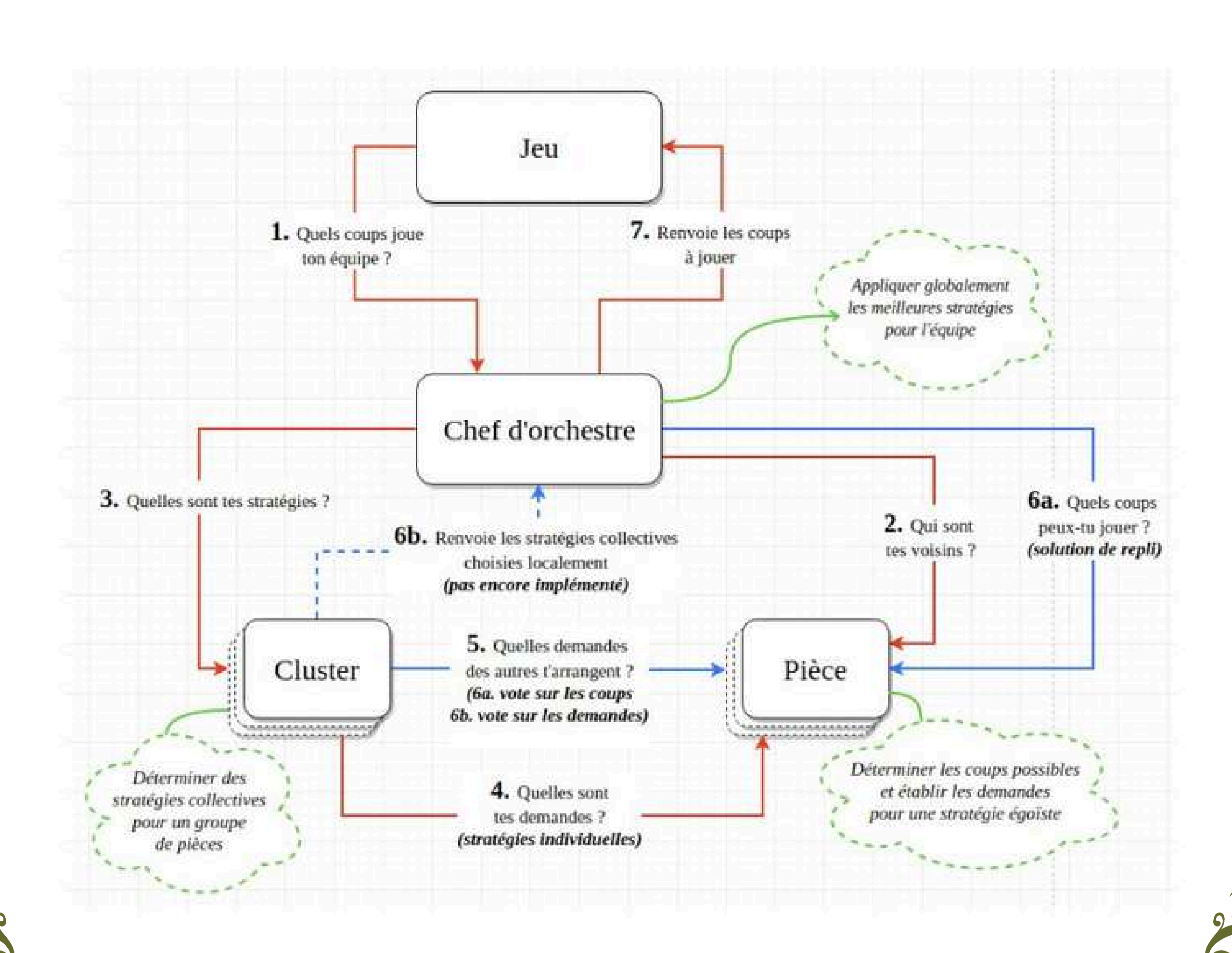


Checkshake (v0.4.2): Modélisation distribuée à travers un jeu stratégique (2)

- Mon rôle sur ce projet était de participer à la **conception et l'implémentation du système multi-agent** et **des règles de raisonnement et de communication entre les agents**.
- Également, j'ai participé à la partie **d'interfaçage** et de **visualisation** du déroulement d'une partie entre deux systèmes multi-agents concurrents.

Fonctionnement

- À chaque tour, le chef d'orchestre de l'équipe (Conductor) demande à chaque pièce (Piece/Brain) quels sont ses voisins, afin de constituer les clusters (Cluster).
- Chaque pièce détermine ses coups possibles et établit des requêtes de stratégies égoïstes en se basant sur ses voisins.
- Chaque cluster récolte les requêtes (Request) de ses pièces.
- Chaque cluster demande à ses pièces quels sont les coups (Path) qui les arrangent afin de procéder au vote sur les coups.
- Le chef d'orchestre récupère alors les meilleurs coups possibles et les transmet au jeu (Game).



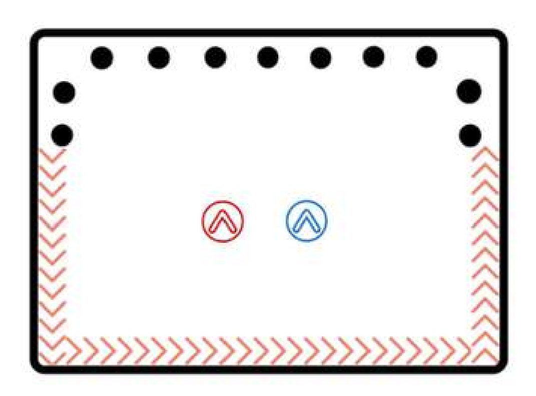
En savoir plus ou contribuer au projet :
https://github.com/Nonudian/CheckShake

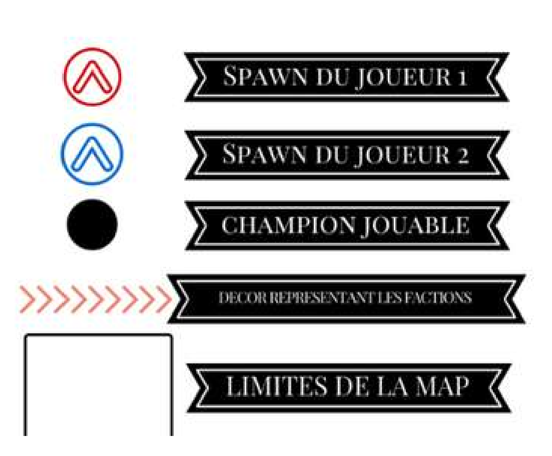
a Wanderer's Adventure : cahier des charges d'un jeu vidéo de versus en tour par tour

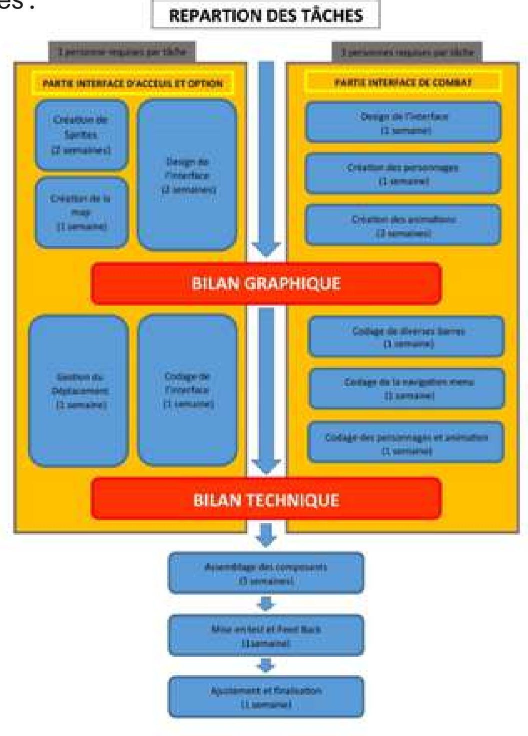
Dans le cadre de ma deuxième année de DUT Informatique, j'ai rédigé un cahier des charges pour un jeu de versus au tour par tour.

Cela a impliqué plusieurs points de compétences développés :

- **Réalisation de maquettes**
- **Planification du projet**
- **Écriture du lore du jeu**







3. Interface de combat :

Une fois les options choisies et validées, notre jeu rentre en **phase de combat**.

L'interface de combat, permettra l'affichage des différents composants de ce qui représente le corps de notre jeu. Seront ainsi projetés : la **modélisation** des deux personnages combattants, les **deux barres d'HP** (Health Points) et de **Mana/Energie**, le **nombre de tours** et le temps restant avant de sauter le tour d'un joueur trop indécis. Sera aussi implémentée, une interface comprenant la **liste des différentes actions effectuables** par le joueur dont c'est le tour **(Sorts, Attaques de base, Objets, Passer son tour (Provocation / Permet de régénérer un peu d'Energie/Mana))**.

Il y aura aussi besoin de réaliser :

-Les méthodes permettant la gestion des différentes compétences de base et **spéciales** (compétences qui rendent la cible vulnérable ou la font passer son tour etc…).

-Les méthodes permettant de **calculer les points de vie/mana/énergie restants**.

-Les **animations** propres à chaque compétence.

-Le **menu de pause** accessible en appuyant sur Echap permettant de revenir à la phase de sélection des combattants, de modifier le volume de la musique d'ambiance du combat, et de quitter le jeu.



a Wanderer's Adventure : cahier des charges d'un jeu vidéo de versus en tour par tour (2)

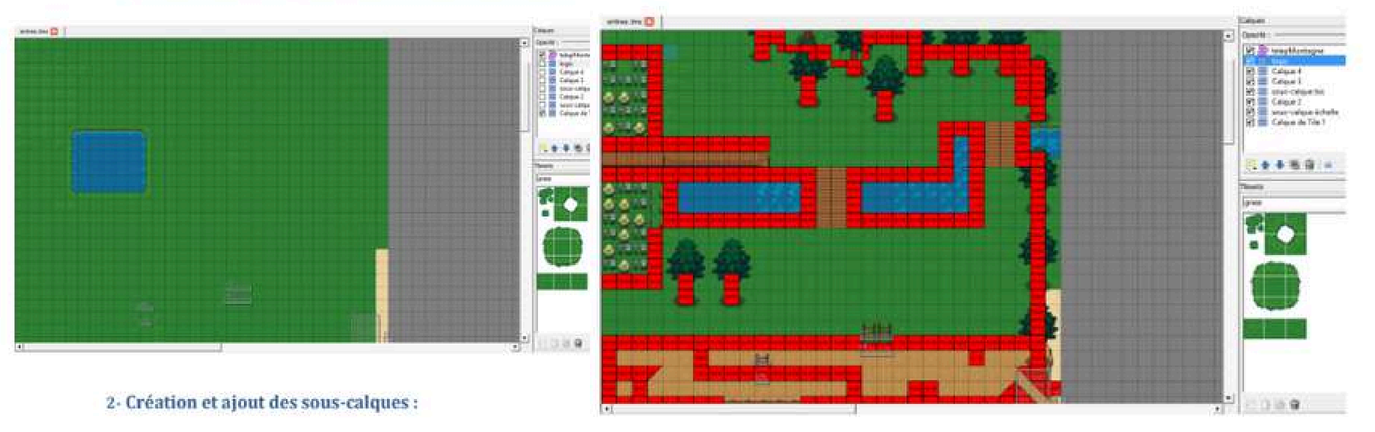
Également, ce projet m'a permis de prendre en main **Slick2D**, une bibliothèque de développement de jeu vidéo 2D en Java ; ainsi que **TiledMap**, un logiciel de réalisation de Cartes de jeu.

Cela m'avait donc permis de m'initier à différentes composantes du développement d'un jeu vidéo :

- **Collection d'assets**
- **Réalisation d'une carte avec plusieurs calques**
- **Animation d'un personnage en 2D**
- **Gestion des contrôles, de la caméra, des collisions et des triggers**

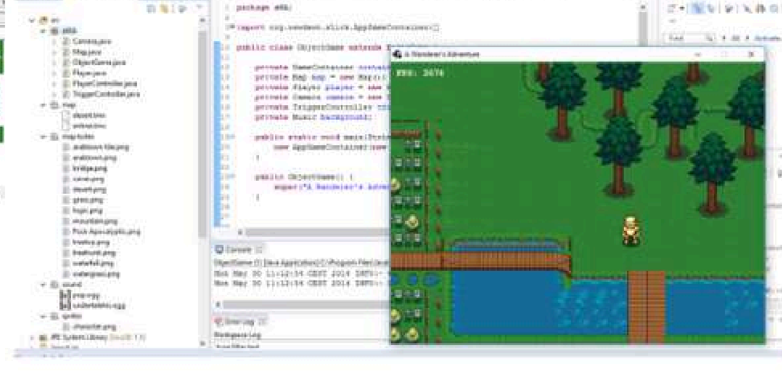
1- Création du calque principal de la Map de notre démo :

3- Ajout d'un calque « logic » qui permet de définir les collisions :

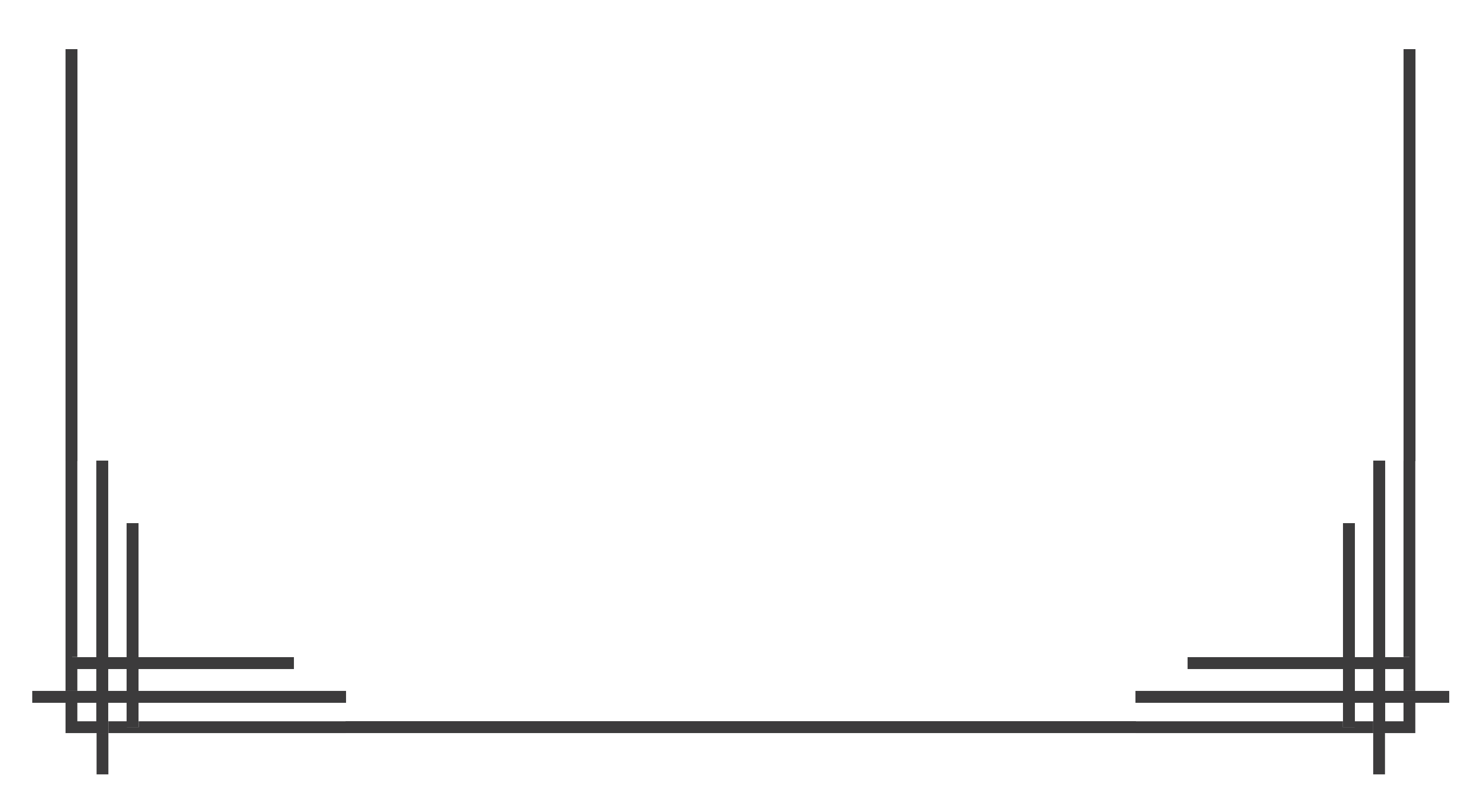


2- Création et ajout des sous-calques :

4- Rendu après ajout des Sprites d'un personnage et codage du jeu grâce aux Classes Slick2D :



AUTOUR DE LA VISUALISATION



Partiggle : Simulateur de particules en C++

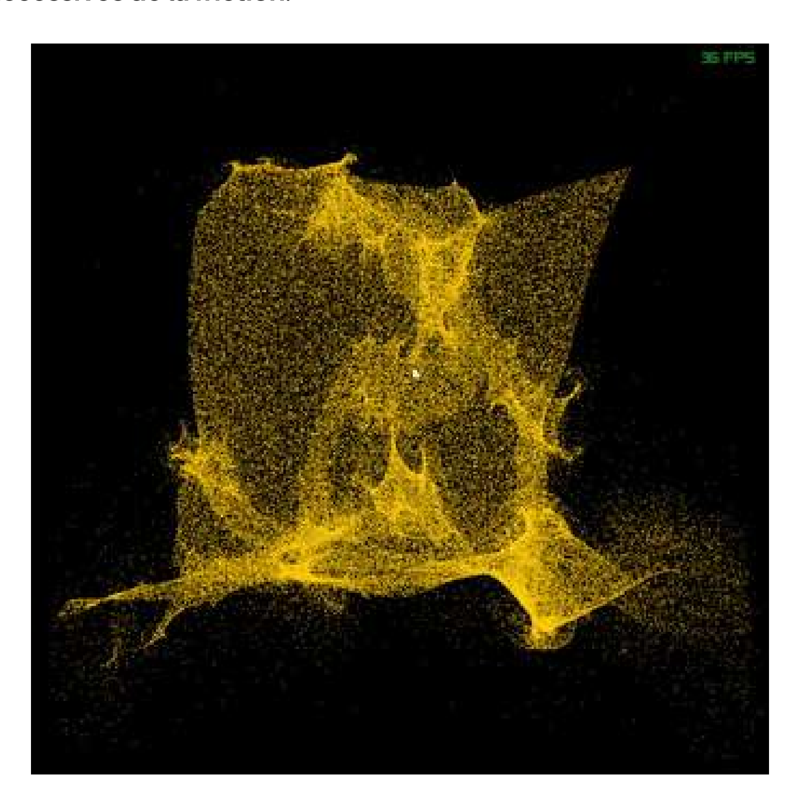
- Dans l'optique d'en découvrir encore plus sur les possibilités de la bibliothèque Raylib, j'ai réalisé ce **simulateur de particules** où le joueur peut **cliquer avec le pointeur de sa souris** afin d'**attirer les particules générées vers le pointeur**.
- Les particules ont également une vélocité qu'elles conservent même quand le clic de la souris est relâché.

Technologies utilisées :

- C++14
- Librairie Raylib pour le développement de jeu vidéo

Règles implémentées :

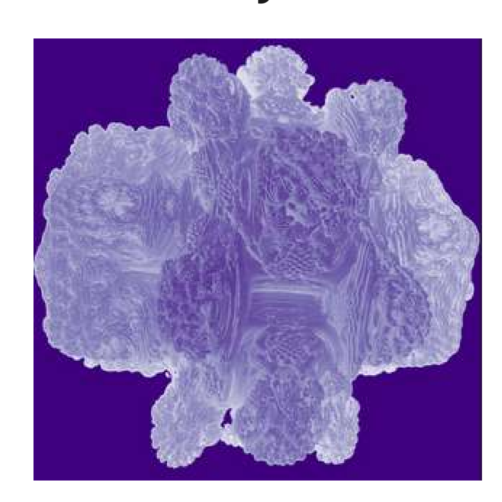
- Les particules font la taille d'un **pixel**
- Elles sont initialisées avec **une position et une vélocité initiale aléatoire**.
- Quand l'utilisateur clique sur le **bouton gauche** de sa souris dans la fenêtre de jeu, les particules, un **vecteur de déplacement s'applique sur les particules vers le pointeur de la souris**.
- Lors des déplacements, **une friction s'applique** sur les particules.
- Le vecteur vitesse des particules **demeure après relâchement** du clic et **s'estompe avec les applications successives de la friction**.



En savoir plus ou contribuer au projet :
https://github.com/NajibXY/Particle-Gaggle-using-CPP-and-Raylib

Génération et rendu en ray marching d'un Mandelbulb en C++ et Python

- Afin de m'exercer et d'en apprendre davantage sur les mathématiques derrière la génération et le rendu des fractales, j'ai travaillé sur cet exemple de **génération de Mandelbulb**.
- Le Mandelbulb est un **type de fractale 3D** émergeant des expérimentations visant à trouver une **représentation canonique 3D** de l'ensemble de Mandelbrot.

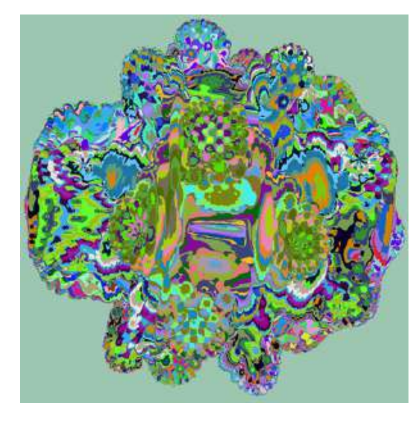


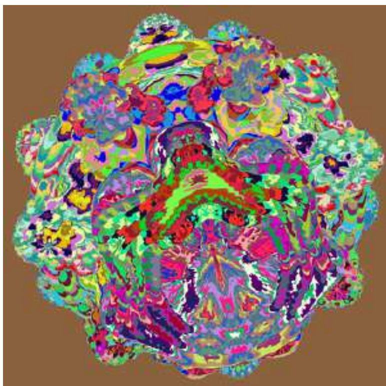


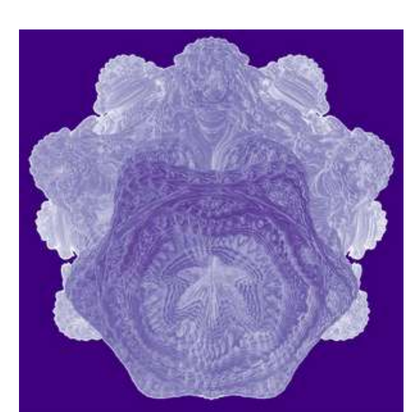
Technologies utilisées :
- C++14, OpenMP
- Python, Numpy, Matplotlib, FFMPEG

Génération des données :
- Les points de données du Mandelbulb sont assez **faciles à générer**.
- Mais pour pouvoir afficher ces données, une méthode de **ray marching** a été implémentée, qui implique **le calcul des lumières et des distances selon le mouvement d'une caméra virtuelle**. Cela nécessite de faire de la **parallélisation** avec OpenMP et **beaucoup d'algèbre vectorielle tridimensionnelle** pour obtenir un temps d'exécution acceptable.
- Il faut environ 30 minutes pour générer les données avec un exposant de base égal à 7.

Exemples d'affichage des données de rendu grâce au script Python :







En savoir plus ou contribuer au projet :
https://github.com/NajibXY/Mandelbulb-with-Ray-marching

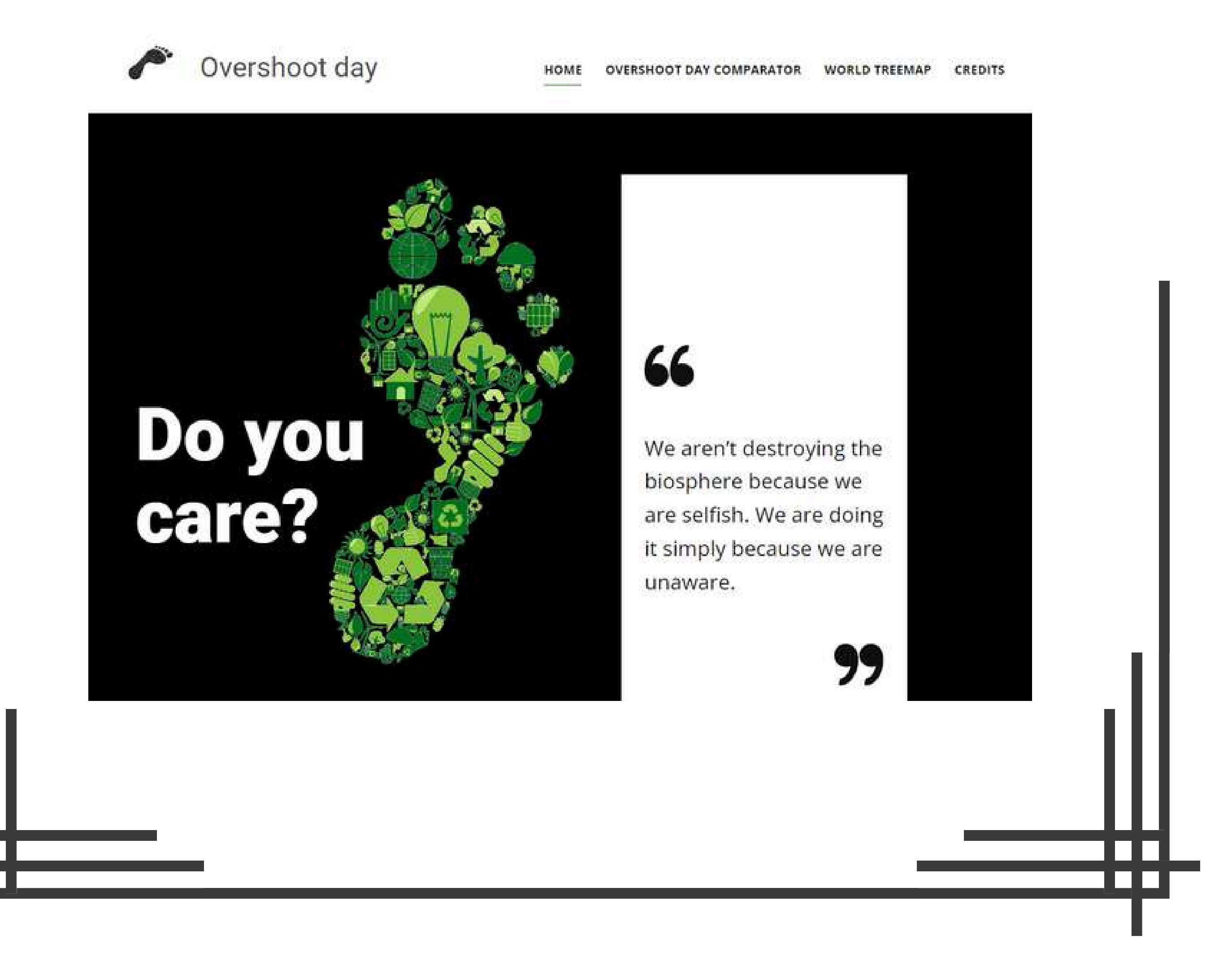
OverShootDay : une visualisation interactive alertant sur la surconsommation

"Le jour du dépassement, ou jour du dépassement de la Terre (en anglais : Earth Overshoot Day ou EOD), correspond à la date de l'année, calculée par l'ONG américaine Global Footprint Network, à partir de laquelle l'humanité est supposée avoir consommé l'ensemble des ressources naturelles que la planète est capable de produire en un an pour régénérer ses consommations ou absorber les déchets produits, dont le dioxyde de carbone. "

Tel est défini le jour du dépassement (Over shoot day en anglais) par Wikipédia.

- Le but de ce projet réalisé dans le cadre de ma deuxième année de Master était d'**alerter** sur le fait qu'année après année, **le jour du dépassement** ne fait que de se **rapprocher du début de l'année dans la majorité écrasante des pays du monde.**

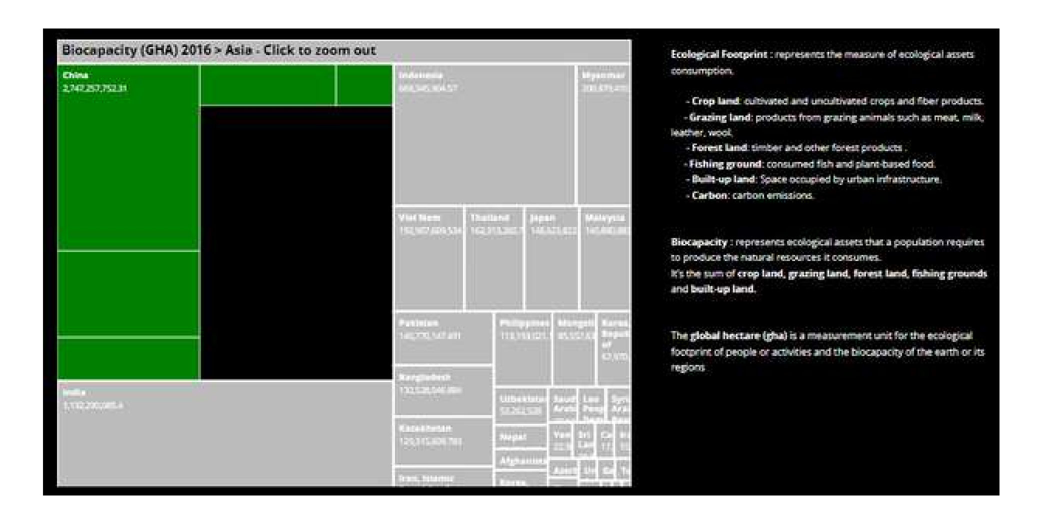
Technologies utilisées :
- HTML/CSS
- Javascript
- Librairie D3.js pour la visualisation de données



OverShootDay : une visualisation interactive alertant sur la surconsommation (2)

Réalisations :

- Avec Enma Barred, Antoine Poloce et Nabil Lamrabet, nous avons **collecté les données de consommation, d'empreinte carbone et de biocapacité des 50 dernières années** afin de permettre la visualisation de ces données.
- Un des onglets permet donc de **visualiser sous la forme d'une TreeMap la biocapacité et l'empreinte carbone des pays du monde**.



- Mon rôle était de **développer le comparateur de l'évolution de jour du dépassement par pays** :

Select a Country : (will be added to the plot and to the legend below)
Morocco
Add to plot
Qatar
France
Morocco

Past Overshoot Days per Cou
900
800
700
600
500
400
300
200
100
0
1965 1970 1975 1980 1985 1990 1995 2000 2005 2010 2015
Qatar France Morocco

Sources du projet :
https://github.com/NajibXY/Overshootday-backup



AUTOUR DE LA RECHERCHE



Mémoire de stage de recherche :

Identification et comparaison d'algorithmes d'apprentissage par renforcement multi-agent appliqués à un contexte éthique

- Dans le cadre de mon stage de recherche au laboratoire LIRIS de Villeurbanne, j'ai réalisé un **mémoire sur la comparaison d'algorithmes d'apprentissage par renforcement multi-agent profond appliqués à un contexte éthique : celui des Smart Grids**
- Cela m'a permis d'acquérir une certaine **aisance** concernant l'**analyse d'articles de recherche** et de **développer mes compétences en Python** et en l'utilisation de la **bibliothèque PyTorch.**

Extrait de mon mémoire de stage :

"***Abstract :*** *La forte croissance du domaine de l'Intelligence Artificielle a entraîné le développement d'algorithmes destinés à des applications de plus en plus complexes, où des contraintes morales et éthiques peuvent faire face à des besoins de performance et d'efficacité. Les algorithmes et les expérimentations sont nombreux, mais nous nous intéressons ici à la simulation de grilles électriques intelligentes (Smart Grids) par le biais d'un système multi-agent. Plus particulièrement, nous adressons l'apprentissage des agents de cette simulation grâce à des algorithmes d'apprentissage par renforcement, les freins rencontrés par ce mécanisme d'apprentissage, ainsi que les questionnements éthiques soulevés par cette simulation. En premier lieu, nous identifions les problématiques sous-jacentes à ce cas de figure ; ensuite nous passons en revue les différentes approches algorithmiques présentes dans la littérature, en portant une attention particulière à celles utilisant des techniques récentes d'apprentissage profond ; puis nous proposons les approches qui nous semblent les plus adaptées à notre cas d'application éthique ; avant de conclure sur une étude comparative de certaines de ces approches.*
Mots-clés : *Intelligence Artificielle, Systèmes Multi-Agents, Apprentissage profond, Apprentissage par Renforcement, Réseaux de distribution intelligents, Éthique, Non-stationnarité.*"

Accéder à la suite du rapport :

https://github.com/NajibXY/Articles-backup/blob/main/M2IA_Stages_2021_paper_21.pdf



Projet d'orientation de master :

Refactoring et optimisation d'un projet de recherche sur les suites de Büchi

- Dans le cadre de mon **projet d'orientation** de Master, j'ai pu contribuer à un projet de recherche alliant **mathématiques avancées et informatique**.
- Cela a été une excellente introduction aux **méthodes de la recherche scientifique et à la collaboration dans des contextes de R&D réels**.
- Les chercheurs Pablo SÀEZ, Xavier VIDAUX et Fabrice JAILLET cherchent à caractériser les suites de Büchi entières de longueur 4. Le sujet associe logique mathématique et informatique. Afin de **confirmer** leurs différentes hypothèses, ils ont **besoin d'un grand nombre de suites** sur lesquelles s'appuyer. Tristan GUILLARD et moi-même nous sommes particulièrement penchés sur la **génération de suites de longueur 4 grâce à une méthode matricielle**.
- Cela nous a permis de **développer nos compétences en C++ et particulièrement l'usage d'OpenMP et Boost**, ainsi qu'en **Python et ses différentes bibliothèques scientifiques : sci-kit, numpy** etc.

<u>Extrait de notre rapport de projet :</u>

2.1 Suites de Büchi

Une suite de Büchi $(x_i)_{i\in\mathbb{N}} \in \mathbb{N}^M$ satisfait l'équation suivante

$$\forall i < M-3, \quad x_{i+2}^2 - 2x_{i+1}^2 + x_i^2 = 2$$

On remarque que toute suite de la forme $(x, \pm(x+1), \pm(x+2), ...)$ ou $(x, \pm(x-1), \pm(x-2), ...)$ respecte cette équation (peu importe la longueur de la suite). On nomme ces suites les suites TRIVIALES. Elles seront utiles pour la méthode des matrices, mais ce ne sont pas celles que nous cherchons à caractériser ou à calculer. Les suites de longueur 4 ont par ailleurs déjà été caractérisées dans $\mathbb{Q}$.

" ***Abstract :*** *Les suites de Büchi sont des suites mathématiques aux propriétés simples mais dont certaines ne sont pas encore complètement comprises. Les suites de Büchi entières de longueur 3 ont déjà été caractérisées en 2011. Nous nous sommes joints à Pablo SÀEZ, Xavier VIDAUX et Fabrice JAILLET, une équipe qui cherche à caractériser les suites de Büchi réelles de longueur 4. L'un des axes de leurs travaux consiste à générer de nombreuses suites réelles de longueur 3 pour les étendre, si elles peuvent l'être. Ils utilisent ensuite ces suites afin de trouver des caractérisations, notamment polynomiales, des suites réelles de longueur 4.*

Nous avons rejoint le projet pour aider les chercheurs dans la génération de ces suites. Notre but principal a été d'accélérer leurs programmes de calcul des suites, notamment grâce à une méthode de génération des suites par des calculs matriciels spécifiques qui génère des suites de Büchi de longueur 3.

Grâce à des études de profiling, des optimisations de logique de code, un portage en c++ et à la libraire boost (multiprecision et numeric::ublas), les programmes ont pu être accélérés d'un facteur 500. De plus, cette méthode permet d'aller chercher des suites très grandes en valeur (jusqu'à 10^10000 sans baisse de performances), là où les méthodes actuelles ne leur ont pas permis d'accéder à des suites d'ordre de grandeur supérieur à 10^20. Cependant, les suites de Büchi de longueur 4 deviennent de plus en plus rares dans les grandes valeurs, ce qui a limité l'exploitation dans le temps du POM. Mais les résultats sont très intéressants et feront l'objet d'une analyse plus poussée. De plus, les outils sont laissés à la disponibilité des chercheurs pour leurs futurs travaux."

<u>Accéder à la suite du rapport :</u>

<u>https://github.com/NajibXY/Articles-backup/blob/main/UE-INF1097M_Rapport_.pdf</u>

Implémentation d'un algorithme de croissance de région

- Dans le cadre de ma première année de Master pour l'UE d'Analyse d'image, nous avons été amenés avec Tristan Guillard à **implémenter un Germ Growing Algorithm** (algorithme de génération de croissance) pour la segmentation d'images.
- Nous avons réussi à proposer un **algorithme efficace**, ce qui a entraîné la rédaction d'un **rapport accessible en bas de page**.

Extrait de notre rapport de projet :

"***Abstract*** : *Ce document est le rapport du laboratoire d'analyse d'images MIF17. L'objectif de ce laboratoire était de mettre en œuvre un algorithme de croissance de germes. Une structure de données Germ a été implémentée à l'aide de différents vecteurs et files de la bibliothèque STD. Les graines sont plantées de manière pseudo-aléatoire et la croissance des germes se produit de manière pseudo-parallèle (chaque germe effectue un tour l'un après l'autre) puis fusionne pour effectuer une segmentation d'image satisfaisante d'une image en niveaux de gris. Quelques paramètres peuvent être facilement modifiés en cours de route, tels que le seuil, le nombre de germes, si la valeur des germes sera modifiée ou statique et le voisinage (4 ou 8).*"

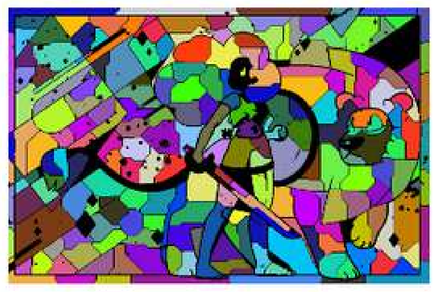

Fig. 5. unfused germs after growth on image 2


Fig. 8. fused 22500 germs on image 2

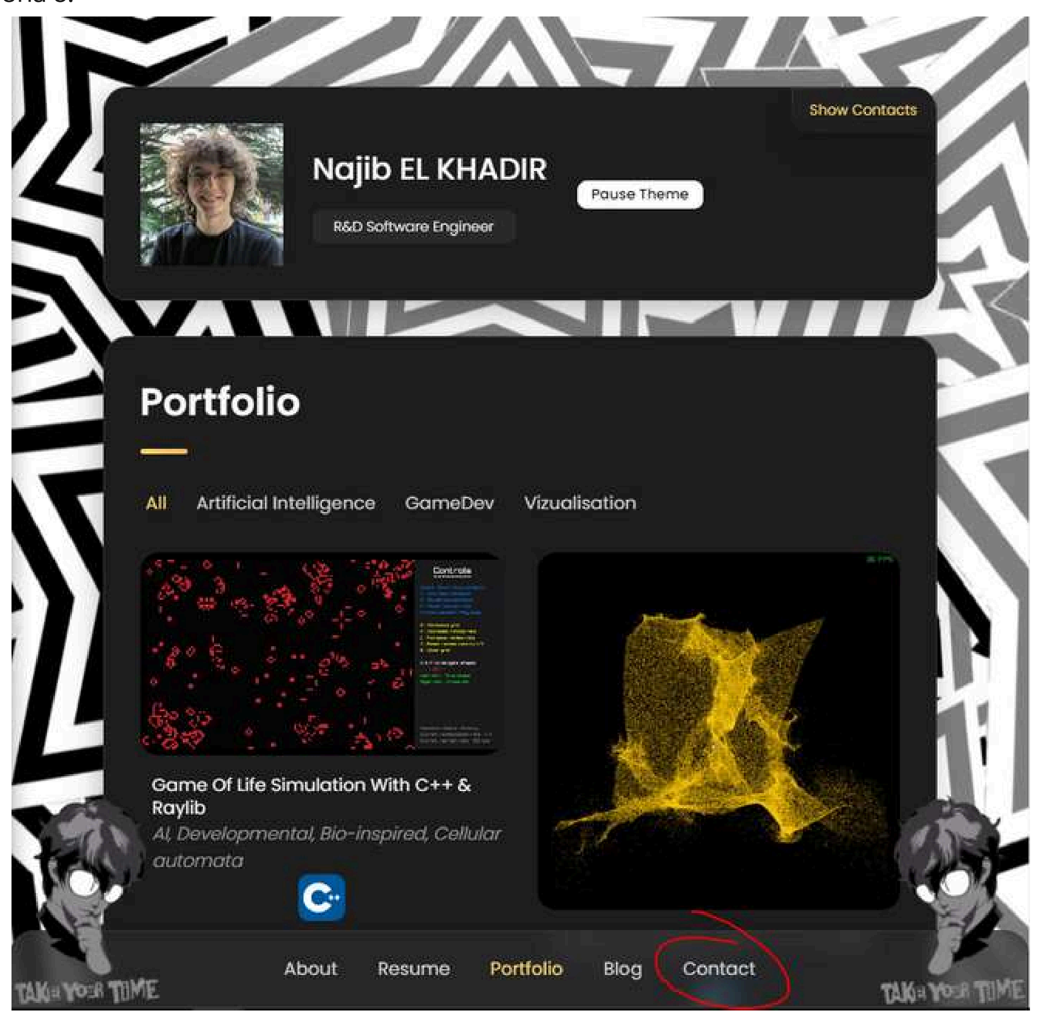
Accéder à la suite du rapport :

https://github.com/NajibXY/Articles-backup/blob/main/rapport_analyse_image.pdf

ME CONTACTER

Portfolio Web : ***najibxy.github.io***

- Afin de me contacter, il est possible de passer par le portfolio version web que j'ai réalisé — onglet "Contact".
- Ce portfolio est ma vitrine professionnelle et me sert à répertorier mes projets et présenter mes réalisations.
- Le site web est totalement responsive et j'ai choisi de rajouter un thème d'un de mes jeux vidéo préféré : Persona 5.



- Il est également possible de me joindre sur
  - LinkedIn : ***linkedin.com/in/najib-el-khadir-403b49b8/***
  - GitHub : ***github.com/NajibXY***

Ainsi que par mail à ***naj.elkhadir@gmail.com*** ou au téléphone au +**33617426917**